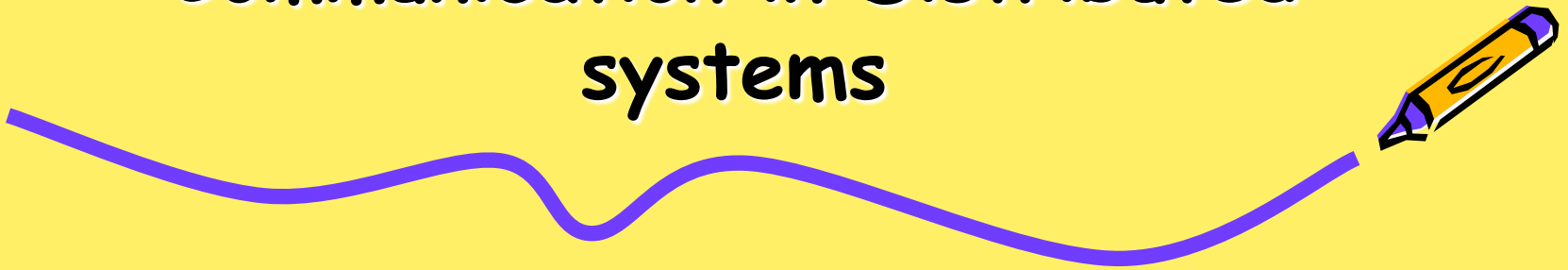


Chapter 2

Communication in Distributed systems



Introduction

- The single most important difference between a distributed system and a uniprocessor system is the interposes communication.
- In a uniprocessor system, most interprocess communication implicitly assumes the existence of shared memory.
- In a distributed system there is no shared memory whatsoever, so the entire nature of interprocess communication must be completely rethought from scratch.

Interprocessor Communication

- Where there are N processors each with its own individual data stream i.e. SIMD and MIMD, it is usually necessary to communicate data between processors.

This is done in two ways

1. Using SHARED MEMORY and SHARED VARIABLES or
2. via an INTERCONNECTION NETWORK

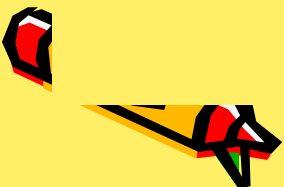


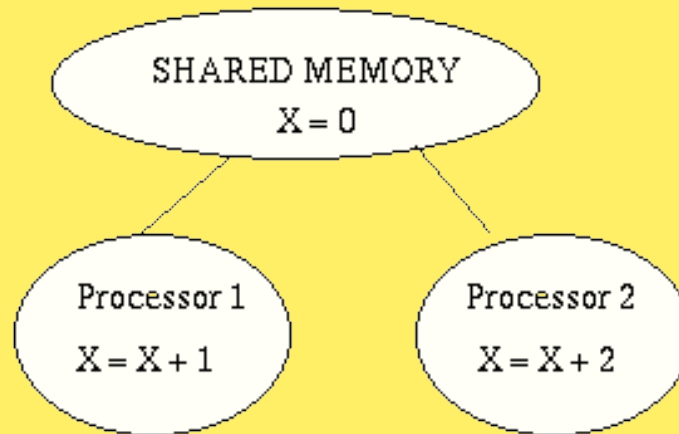
Fundamentals of Interprocessor Communication

- Where there are N processors each with its own individual data stream i.e. SIMD. and MIMD. , it is usually necessary to communicate data / results between processors. This can be done in two main ways.

1. Using a SHARED MEMORY and SHARED VARIABLES

- This consists of a global address space which is accessible by all N processors. A processor can communicate to another by writing into the global memory where the second processor can read it.
- Shared memory solves the interprocessor communication problem but introduces the problem of simultaneous accessing of the same location in the memory. Consider.



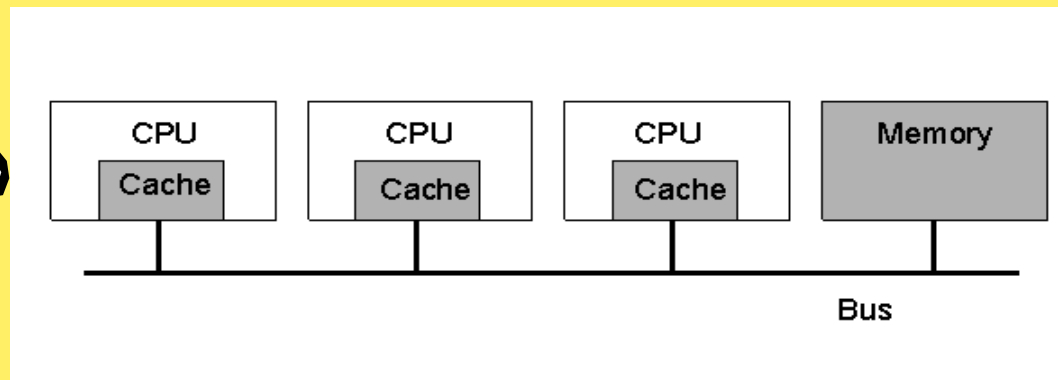


- If x is a shared variable accessible by P1 and P2. Depending on certain factors, $x=1$ or $x=2$ or $x=3$.
- if P1 executes and completes $x=x+1$ before P2 reads the value of x from memory then $x=3$ similarly if P2 executes and completes $x=x+2$ before P1 reads the value of x from memory then $x=3$
- if P1 and P2 read the value of x before either has updated it then the processor which finishes last will determine the value of x .
- if P1 finishes last the value is $x=1$
- if P2 finishes last the value is $x=2$
- In a multiuser, real time environment the processor which finishes last would vary from run to run - so the final value would vary.(NON-DETERMINANCY)



Shared Memory

- Here each processor has access to any variable residing in the shared memory.
- So if processor x wishes to pass a number to processor y it must be done in two steps:-
 - 1- **Processor x** writes the number into the shared memory at a given location accessible to **processor y**,
 - 2- **Processor y** then reads the number from that location.
- During the execution of a parallel algorithm, the N processors can access shared memory for reading / writing data and / or results.



- All processors can gain access to the shared memory simultaneously if the memory locations they are trying to read from or write into are different.
- However we can get problems when two or more processors require access to the same memory location simultaneously.

Example:

- Let x be a variable that can be accessed by two processors $P1$ and $P2$.
- Now consider the assignment $x := x + 1$, which is normally done in three stages.
 1. copy value of x into some register
 2. add 1 to value on register
 3. store value on register at the address for x
- Say now that $P1$ and $P2$ both execute such an assignment, assume $x=0$ initially
- What is the final result ?



- P1 copies value of $x (= 0)$ into its register
- P2 copies value of $x (= 0)$ into its register
- P1 adds 1 to its register → these two at the same time
- P1 stores value of $x (= 1)$
- P2 adds 1 to its register
- P2 stores value of $x (= 1)$
- Giving a result of 1 rather than 2. This is because P2 reads the value of $x (= 0)$ before P1 has updated it.
- Therefore depending on whether 2 or more processors can gain access to the same memory location simultaneously, we have 4 subclasses of shared memory computers :-
 1. Exclusive Read, Exclusive Write (EREW) SM Computers
 - Access to memory locations is exclusive i.e. no 2 processors are allowed to simultaneously read from or write into the same location.
 2. Concurrent Read, Exclusive Write (CREW) SM Computers
 - Multiple processors are allowed to read from the same location but write is still exclusive. i.e. no 2 processors are allowed to write into the same location simultaneously



3. Exclusive Read, Concurrent Write (ERCW) SM Computers

- Multiple processors are allowed to write into the same memory location but read access remains exclusive.

4. Concurrent Read, Concurrent Write (CRCW) SM Computers

- Both multiple read and multiple write privileges are allowed.
- Conceptually, each of the several processors reading from that location makes a copy of its contents and stores it in its own register (RAM)
- Problems arise however, with concurrent write access.
- If several processors are trying to simultaneously store (potentially different) data at the same address, which of them should succeed ?
i.e. we need a deterministic way of specifying the contents of a memory location after a concurrent write operation.



- Some ways of resolving write conflicts include :-
 1. Assign priorities to the processors and accept value from highest priority processor
 2. All the processors are allowed to write, provided that the quantities they are attempting to store are equal, otherwise access is denied to ALL processors.
 3. The max / min / sum / average of the value is stored (numeric data).
- Despite this, it is only feasible to allow P processors to access P memory locations simultaneously for relatively small P (< 30)
- Usually because the cost of the communication hardware is too high and physical size of the device used for a memory location.
- So shared memory SIMD machines are unrealistic and no commercial machines exist with this design.




- However in MIMD machines, which use much more powerful processors, shared memory systems are in existence which have small numbers of processors (2 - 30).
- To illustrate the theoretical potential of the four different subclasses of shared memory consider the following example.
- We have N processors to search a list $S = \{ L1, L2, \dots L_n \}$ for the index of a given element x. $1 < N \leq n$ assume x may appear several times in the list and any index will do.


ALGORITHM :

Procedure SM search (S, x, k)

Step 1: FOR i=1 to N do in parallel
 read x
 ENDFOR

Step 2: FOR i=1 to N do in parallel
 $S_i := \{L(i-1)n/N + 1, \dots L_n/N\}$
 perform sequential search on sublist
 (returns $K_i = -1$ if not in list or index if it is in the list)
 ENDFOR

 Step 3: FOR i=1 to N do in parallel
 if $K_i > 0$ then $K = K_i$ endif
 ENDFOR

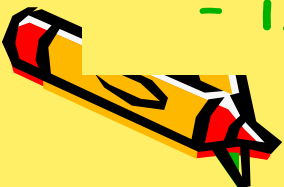
- 
- Assuming the sequential search procedure takes $O(n/N)$ time in the worst case (step 2), what is the time complexity of running this algorithm on the four subclasses of shared memory machine ?

EREW :

- Step 1 takes $O(N)$ time (N reads, one at a time)
- Step 2 takes $O(n/N)$ time (time for reading list & sequential search)
- Step 3 takes $O(N)$ time
 - i.e. overall $O(N) + O(n/N)$ time

ERCW :

- Step 1 takes $O(N)$ time
- Step 2 takes $O(n/N)$ time
- Step 3 takes constant time (with an appropriate conflict resolution rule)
 - i.e. overall $O(N) + O(n/N)$

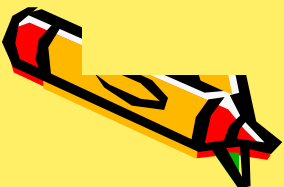


CREW :

- Step 1 takes constant time
- Step 2 takes $O(n/N)$ time
- Step 3 takes $O(N)$ time
 - i.e. overall $O(N) + O(n/N)$

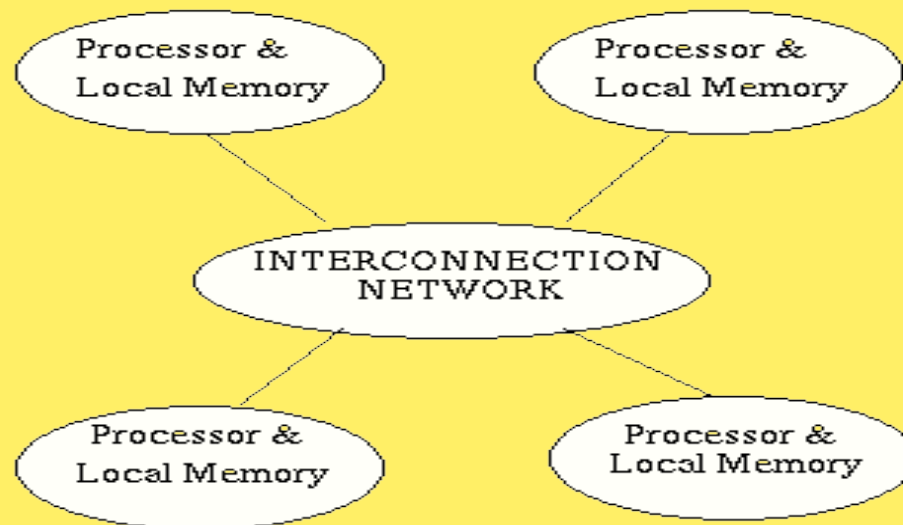
CRCW :

- Step 1 takes constant time
- Step 2 takes $O(n/N)$ time
- Step 3 takes constant time
 - i.e. overall $O(n/N)$



2. INTERCONNECTION NETWORK and MESSAGE PASSING

- Here each processor has its own private (local) memory and there is no global, shared memory.
- Therefore, the processors need to be connected in some way to allow communication of data to take place.

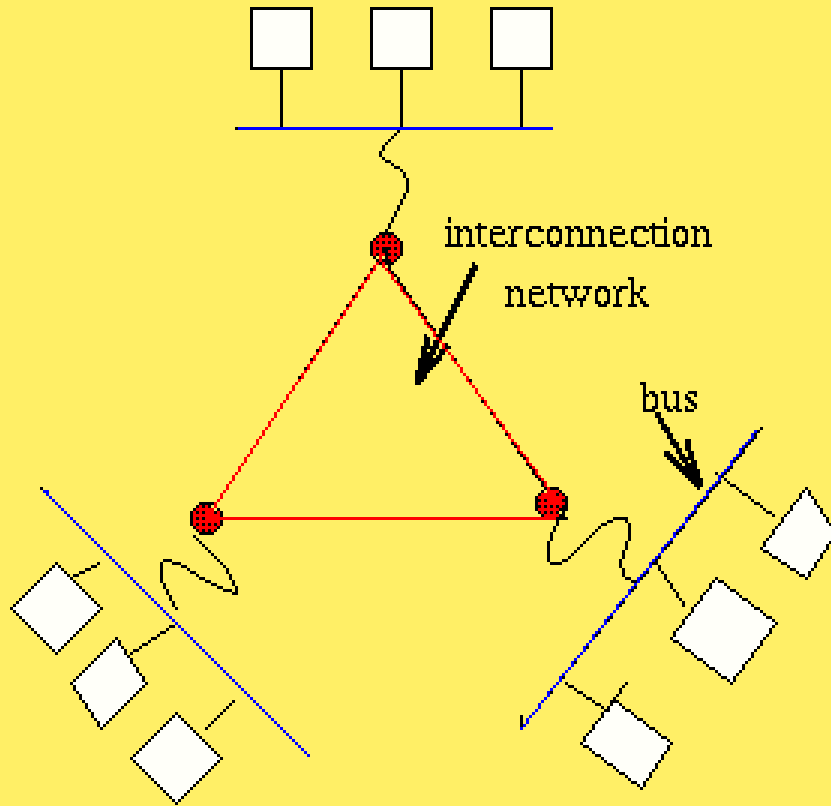


Example:

- if a processor requires data contained in a separate processor then it must be explicitly passed by using instructions supplied for communication e.g. send and receive functions.

• P1	P2
send (x, P1)	receive (x,P2)

- the value of x is explicitly passed from P2 to P1.
- This is known as message passing.
- In addition to the extreme cases of shared memory and distributed memory there are possibilities for hybrid designs that combine features of both.
- E.g. clusters of processeors, where a high speed bus serves for intracluster communication and an interconnection network is used for intercluster communication.



- 
- Example: Summing m numbers using a) Shared memory
b) distributed memory

1- Summing m numbers on a sequential computer we have

- $\text{sum} := A_0$
- FOR $i=1$ TO $m-1$
- $\text{sum} := \text{sum} + A_i$
- ENDFOR
- i.e. $[(A_0 + A_1) + A_2] + A_3 \dots$ etc Giving That $O(m)$ time \rightarrow Is addition inherently sequential ?
- If we have N processors then each processor can calculate the sum of m / N numbers and then the sum of these partial sums will give the final sum.

Using Shared Memory

- The m numbers are held in the global shared memory

Global_sum := 0

FOR all P_i WHERE $0 \leq i \leq N-1$

Local_sum := 0

Calculate local sum of m/N numbers

LOCK

Global_sum := Global_sum + Local_sum

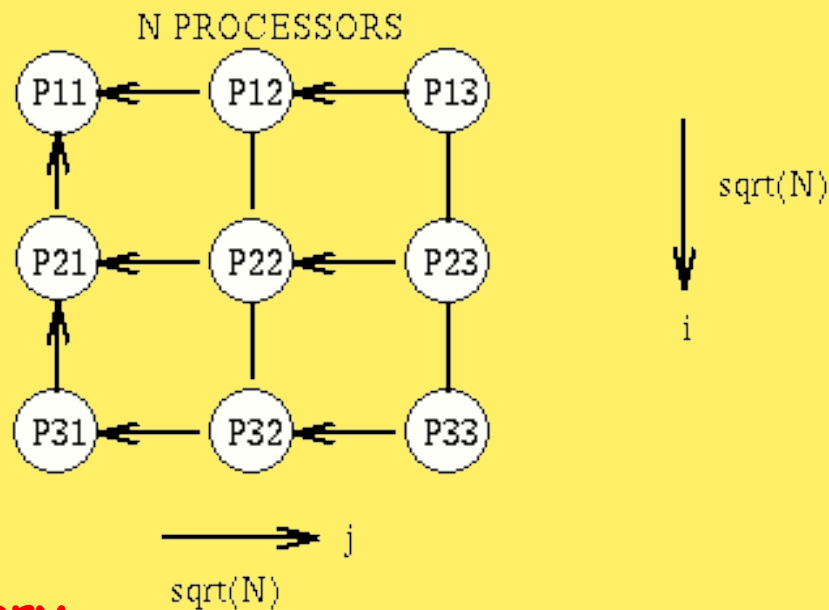
UNLOCK

ENDFOR

- Since `global_sum` is a shared variable each processor must have mutually exclusive access to it - otherwise the final answer will be incorrect.
- Running time (algorithm time complexity) is now $O((m / N) + N) + S$

where:

- m/N comes from adding m/N numbers in parallel
- N comes from adding N numbers in sequence
- S is any time required for synchronization.



Distributed Memory

- Say we have a "square mesh" of processors.
- Each processor finds the local sum of its m / N numbers. Then each processor passes its local sum to another processor (at the correct time) until finally the `global_sum` is contained in processor P11

FORALL P_{ij}

-all processors active

FIND local_sum of m / N numbers

ENDFOR

FORALL P_{ij} $j = \sqrt{N}$ downto 2

- all column j processor

P_{ij} passes local sum to P_{ij-1}





*Pij-1 calculates local sum = new_local_sum +
old_local_sum*

ENDFOR2

FORALL *Pi1* *i = sqrt(N) downto 2* 1 proc. in col *i* active

Pi1 passes local sum to Pi-1,1

local sum = new_local_sum + old_local_sum

ENDFOR

- There are $(\sqrt{N}) - 1 + (\sqrt{N}) - 1$ additions and communications, therefore the total time complexity is $O(m/N + 2(\sqrt{N}) - 2 + C)$
- where: C is the time for communication



Architectural models of multicomputers

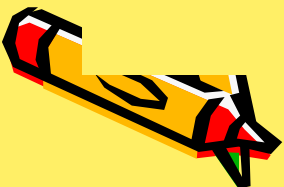
- 1- Client-server model
- 2- Peer-to-Peer model
- 3- variations
 - A- Multiple servers
 - B- Proxy servers
 - C- Mobile code
 - D- Mobile agents
 - E- Thin clients and computers servers
- Each architecture is differs in the following cc/s:
 - The placement of components (data and workload distribution) across the network of computers
 - The interrelationship between the components (the role of each one and pattern of communication)



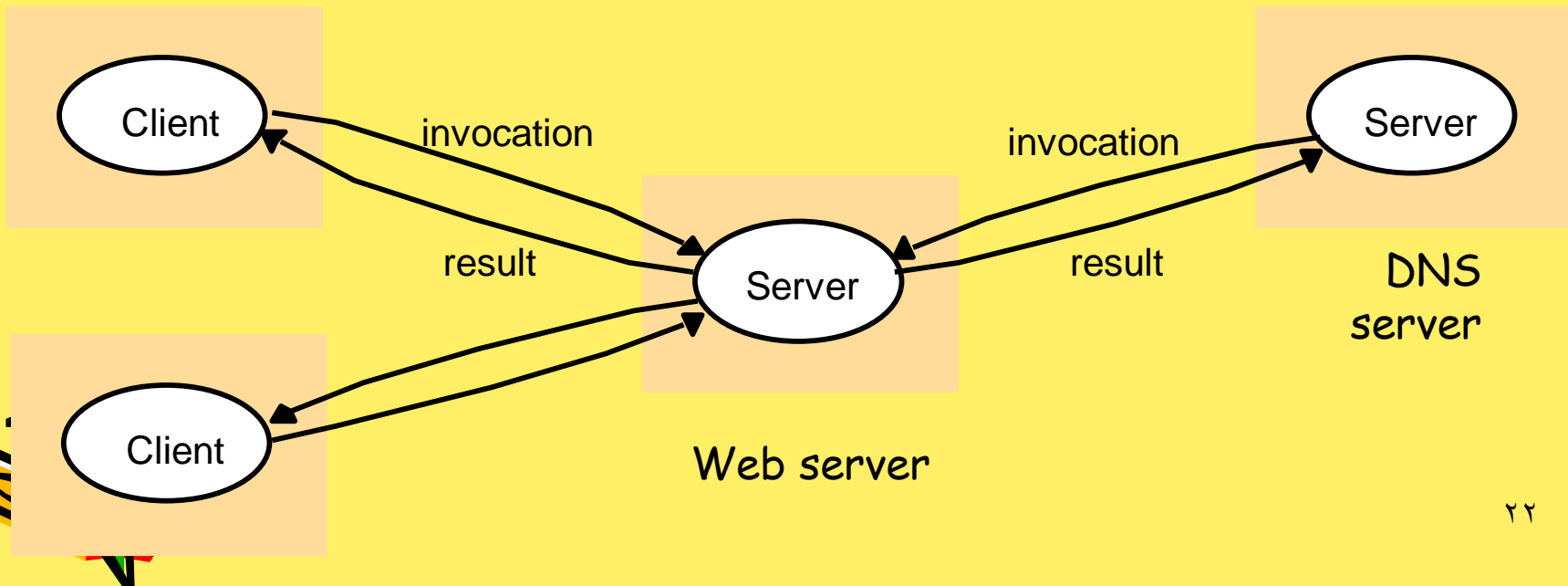
1 - Client/Server Model



- Many distributed systems today are constructed out of interacting clients/servers. Most widely employed (used)
- Server:
 - service software running on a single machine
 - a process on a networked computer that accepts **requests** from other (local or remote) processes to perform a service and **responds** appropriately.
- Client:
 - process that can invoke a service using a set of operations that forms its *client interface*
 - requesting process in the above is referred to as the client.
- Request and response are in the form of messages.
- Client is said to invoke an operation on the server.



- Client process interact with individual server processes in separate host computers to access the shared resources that the hosts manage.
- Servers may be clients for other servers: e.g. a web server is a client of a local file server that manages the files in which the web pages are stored, web servers and most other internet services are clients for the DNS service which translates the Internet Domain Names to network address.





Domain Name System:

- *distributed database implemented in a hierarchy of many name servers*
- *application-layer protocol that is responsible for resolving names (address/name translation)*
- **People:** many identifiers: SSN, name, Passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- Resource “name”, e.g., URL sal.cs.uiuc.edu – human-readable format

Q: given a resource name, how does a client find out the IP address of the service/server?

- Another example, a search engine is a server, but it runs programs called web crawlers that access web servers
- throughout the Internet for information inquiries.



2- Peer 2 Peer Model

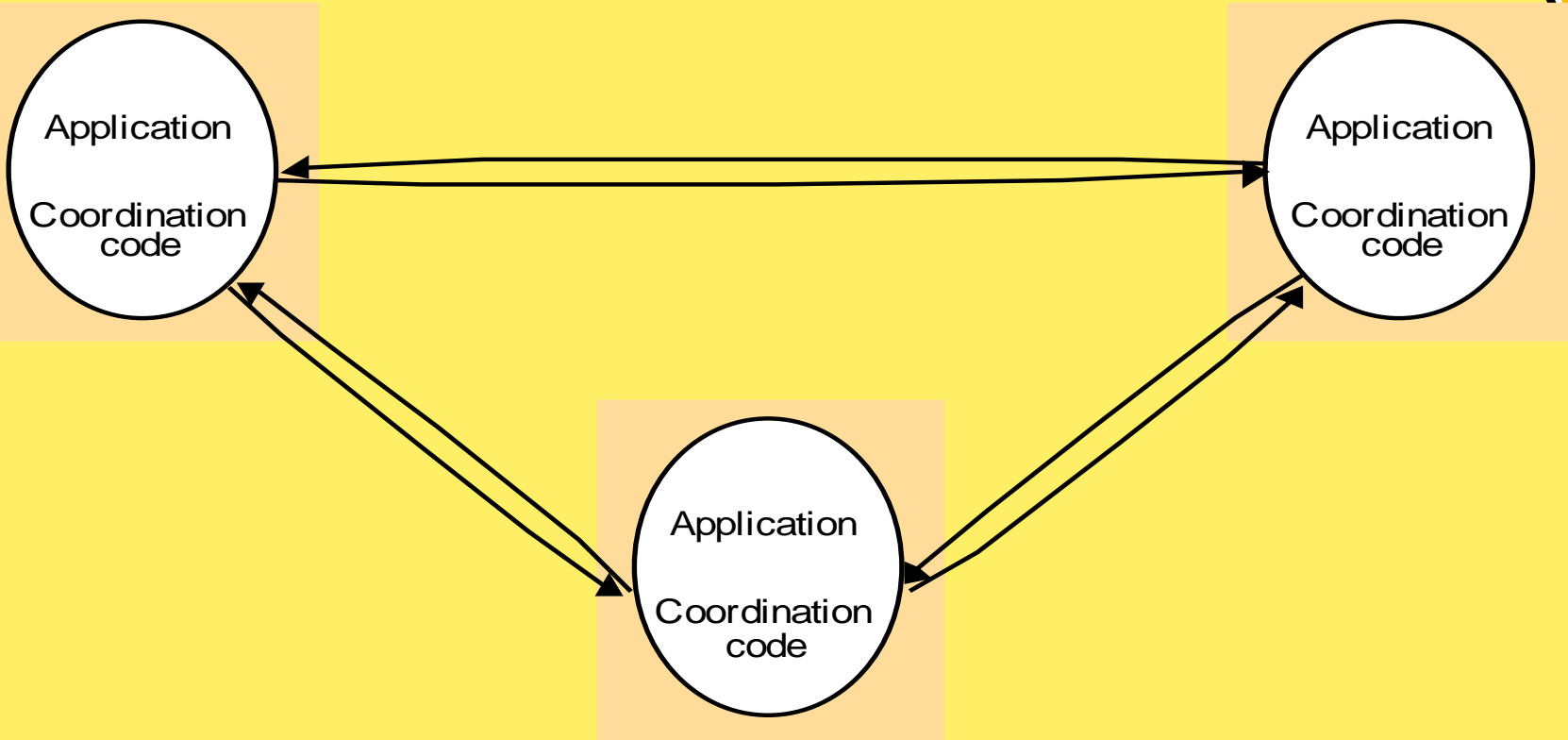


- All the processes play similar roles, interacting cooperatively as peers, without difference between server and client processes or the computer they run on, to perform tasks.
- The problem with client/server is its scale up problem
- Centralizing the services at the server makes it overload under large number of clients
- The overload value is determined by the server's capability and the network bandwidth.
- So, there is a need to distribute shared resources in order to share computing and communication loads to large numbers of computers and network links
- The hardware capacity of today's PC is much larger than yesterday's servers, so we can make use of these PC.



Peer 1

Peer 3



Peer 2

The aim of P2P arch. is to make use of the resources (data or HW) of participating computers to fulfill a given task or activity.

The most famous P2P instance is Napster application.

What is Napster?

- It is an application that facilitates the sharing of music and video recordings among its users around the world.

Client

- Connect to a Napster server
- Upload list of music files that you want to share (names only, not the files themselves!)
 - Server maintains list of
 <filename, ip_address, portnum> tuples

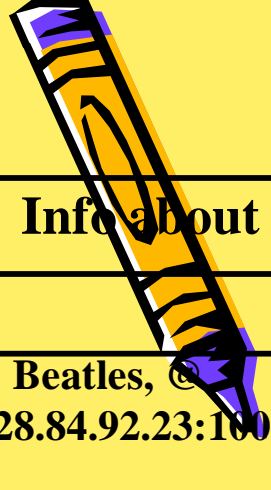
Search from a client:

- Send server keywords to search with
- Server searches its directory with the keywords
- Server returns a list of matching hosts - <ip_address, portnum> tuples to client
- Client pings each host in the list to find transfer rates
- Client fetches file from best host



Napster Structure

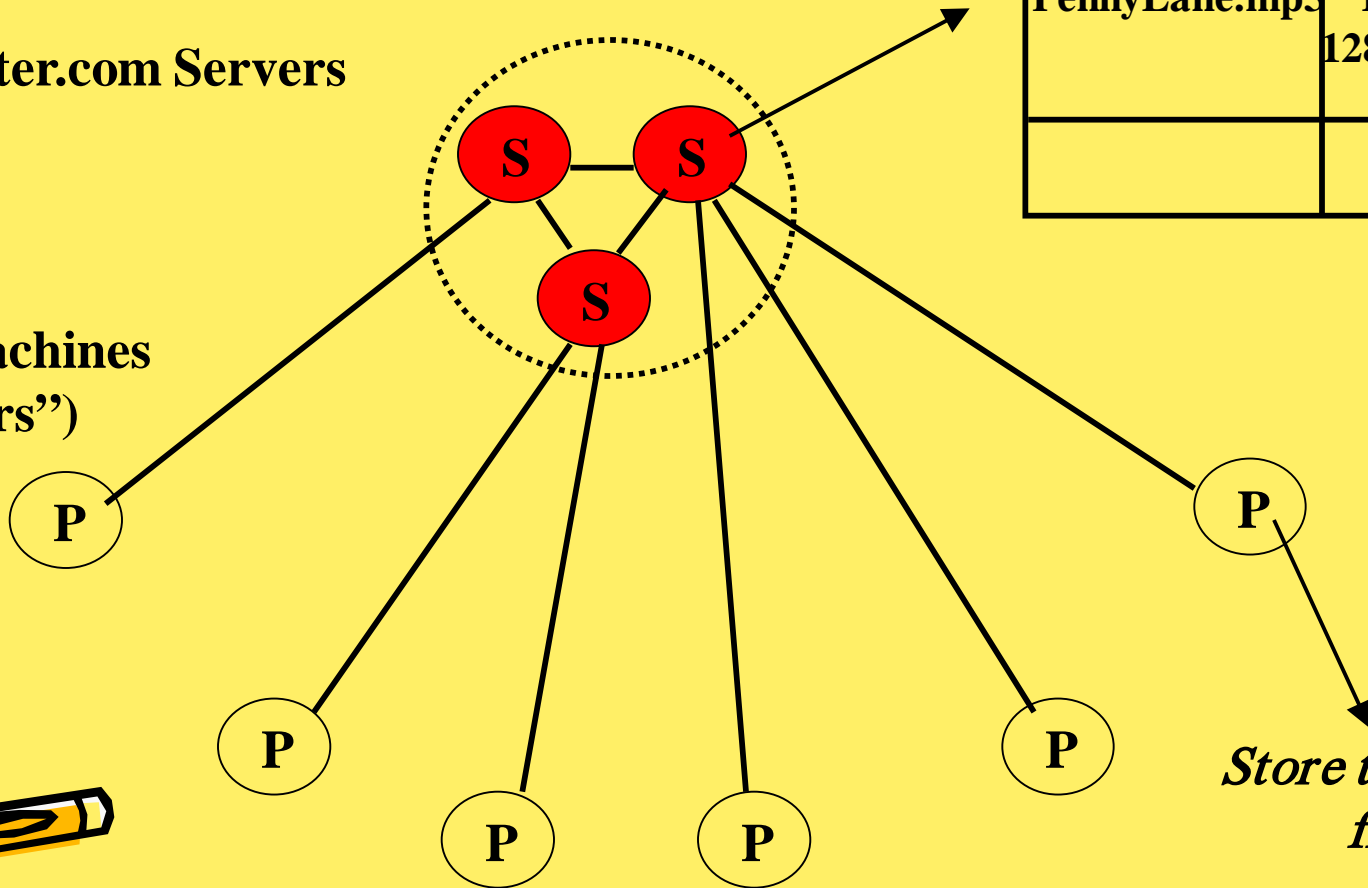
*Store a directory, i.e.,
filenames with peer
pointers*



Filename	Info about
PennyLane.mp3	Beatles, @ 128.84.92.23:1006

napster.com Servers

Client machines
("Peers")



*Store their own
files*



Home



Chat



Library



Search



Hot List



Transfer



Discover



Help

Artist:

Find it!

Title:

Clear Fields

Max Results:

Advanced >>

Filename	Filesize	Bitrate	Freq	Length	User	Connection	Ping
incomplete_other_artist\Tito Puentes Golden Latin Jazz Allstars - Oye Como ...	3,696,640	128	44100	3:51	bdenzler	DSL	343
incomplete_other_artist\[Marty Robbins] The Fastest Gun Around.mp3	542,304	128	44100	0:39	bdenzler	DSL	343
incomplete_other_artist\Ravi Shankar - Chants Of India 04 - Asato Maa.mp3	2,449,408	128	44100	2:35	bdenzler	DSL	343
other_artist\Engelbert Humperdinck - White Christmas.mp3	9,277,648	320	44100	3:52	bdenzler	DSL	343
other_artist\Grateful Dead - Franklin's Tower - Reggae Style.mp3	4,635,458	128	44100	4:48	bdenzler	DSL	343
Unknown Artist - You seriously have to listen to this.mp3	462,848	318	16000	0:17	sam113...	Cable	383
MP3z\artist - 'The Way Life Is' By Drag-On featuring Case.mp3	4,726,784	128	44100	4:54	burg651	Cable	386
MP3z\artist - 'Opposite Of H2O' By Drag-On featuring Jadakiss.mp3	3,540,992	128	44100	3:41	burg651	Cable	386
Various Artist - Perfect Day 97.mp3	3,722,344	128	44100	3:53	falkstad	ISDN-128K	398
Liszt\Liszt - Etude 'Un sospiro' - Czifra-artist.mp3	2,752,512	128	44100	2:53	lskjdfklj...	Unknown	504
Music\Waiting To Exhale - Original Soundtrack Album - Various Artist - Count...	3,199,083	96	44100	4:26	Jzfork9	56K	511
Track 03_artist.mp3	4,054,332	128	44100	4:13	immusic...	Cable	514
Track 02_artist.mp3	6,228,974	128	44100	6:26	immusic...	Cable	514
Track 01_artist.mp3	4,731,426	128	44100	4:54	immusic...	Cable	514
Track 04_artist.mp3	4,514,505	128	44100	4:41	immusic...	Cable	514
Track 05_artist.mp3	4,105,323	128	44100	4:16	immusic...	Cable	514
mixer in track 01_Artist_0721011750.mp3	180,686	128	44100	0:17	immusic...	Cable	514
Album\Reflex - Keep In Touch-Artist.mp3	7,041,024	160	44100	5:49	rotimca	56K	527

Returned 100 results.

Get Selected Songs

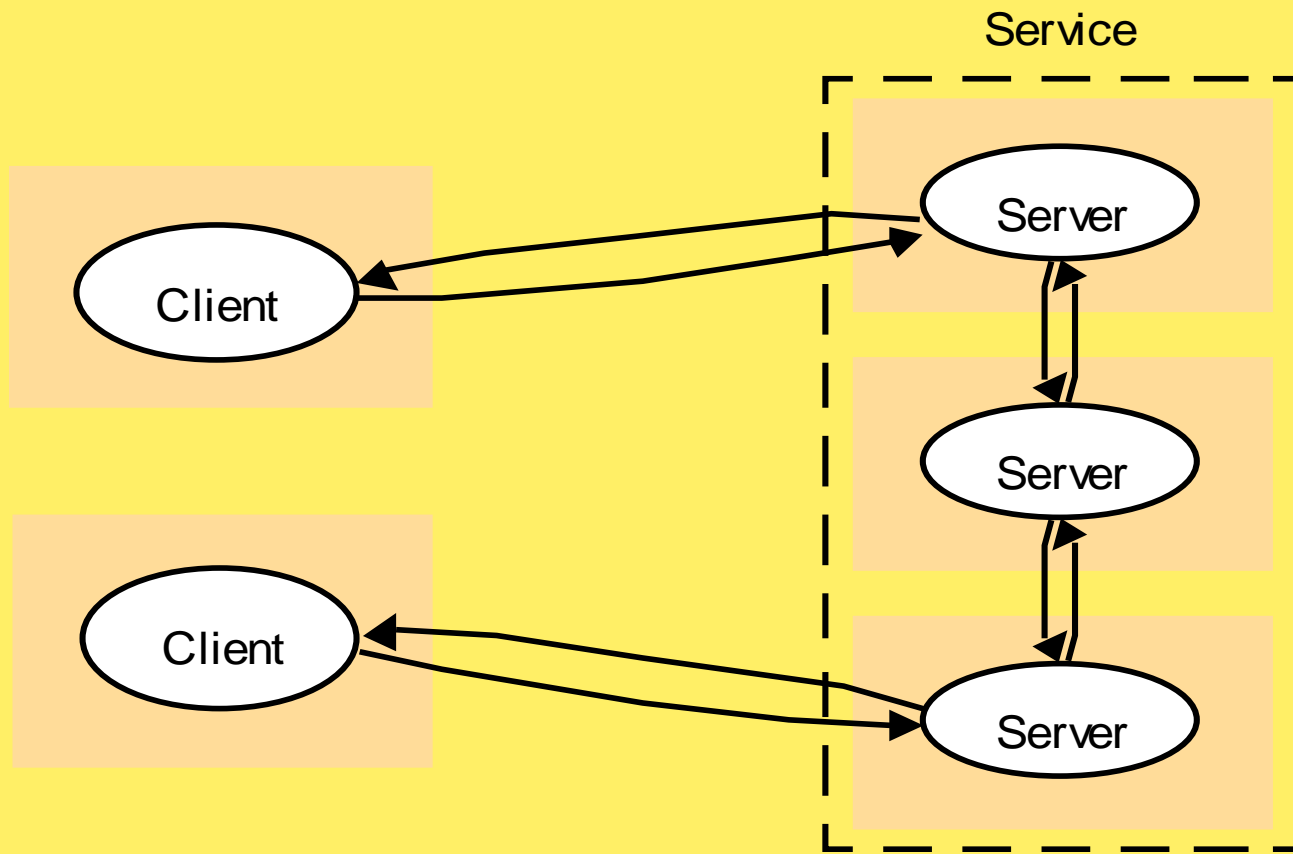
Add Selected User to Hot List

3-A Multiple Servers



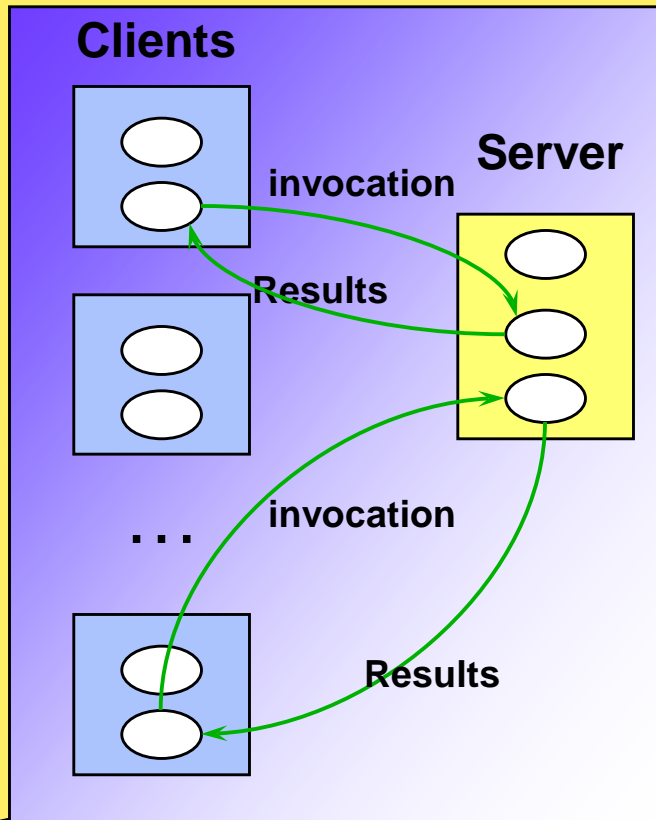
- Services may be implemented as several server processes in separate host computers interacting as necessary to provide service to client processes.
- The server may partition the set of objects on which the service is based and distribute them between themselves or they may make different copies (replicas) of the objects on several host but they have to take care of any changes made to them on any host.
- These two options are illustrated through the following examples:
 - 1- The web is a good example of partitioned data which is distributed on different hosts with each server can manage its own set of data files.
 - 2- The NIS (Network Information Service) is an example of replicated data. NIS is used by computers on a LAN to check users login. A copy (replica) of usernames and passwords is stored on each NIS server.



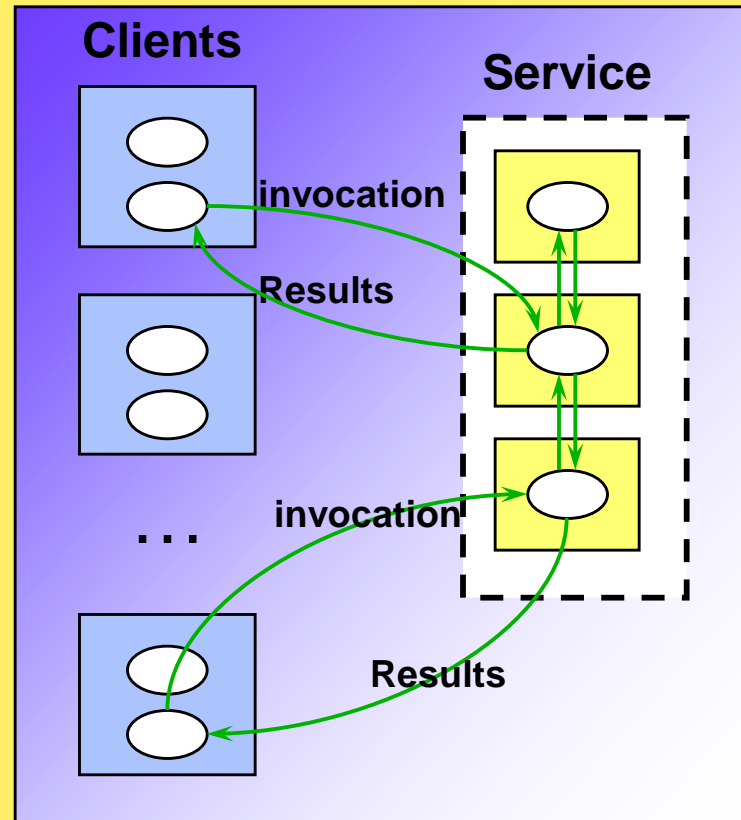


- A cluster is an example of multiple-server architecture which is used for highly scalable web services.
- Service processing can be partitioned or replicated between them.

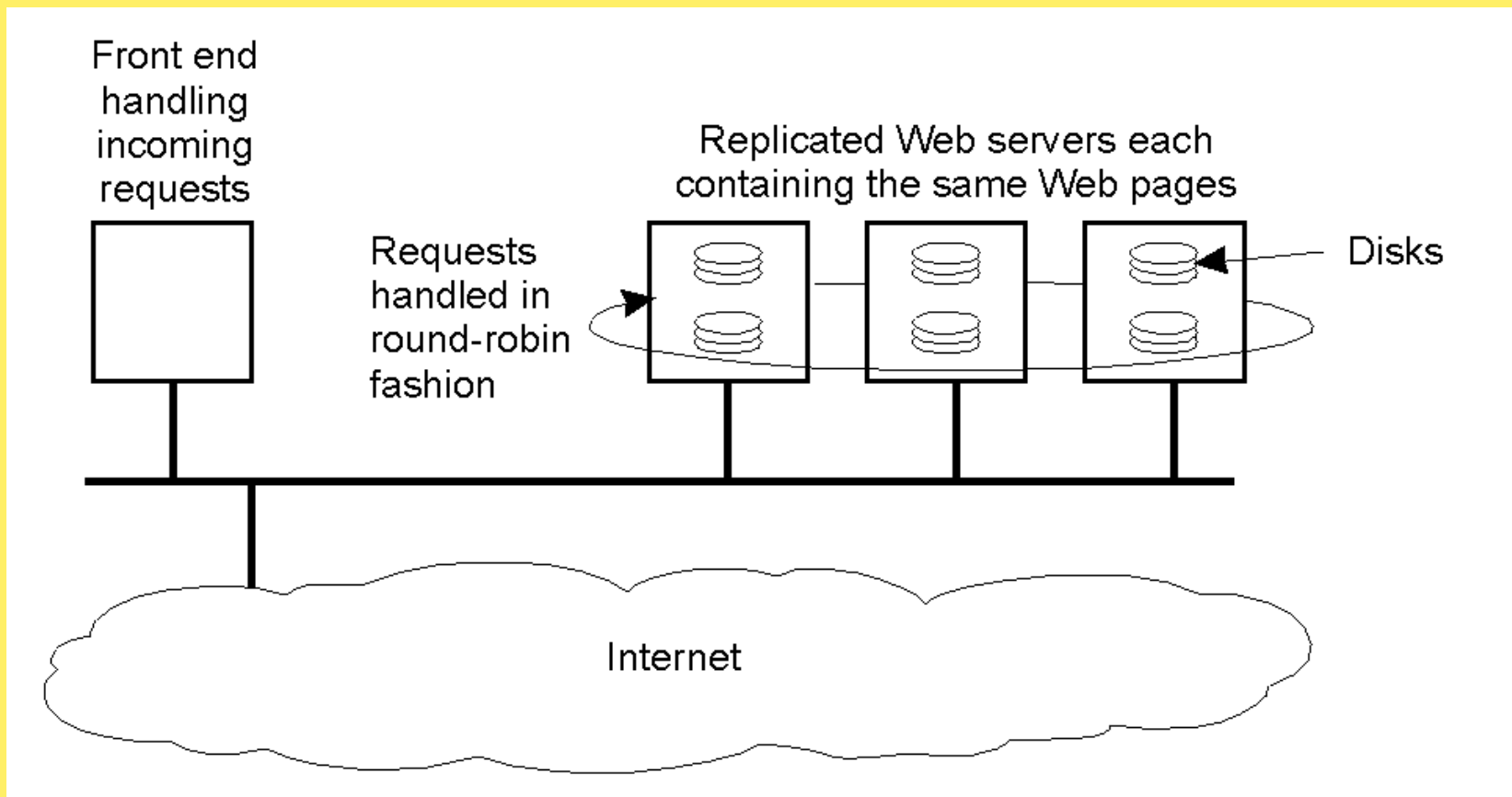
❖ Client-Server Architecture



Single-Server

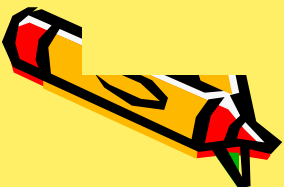


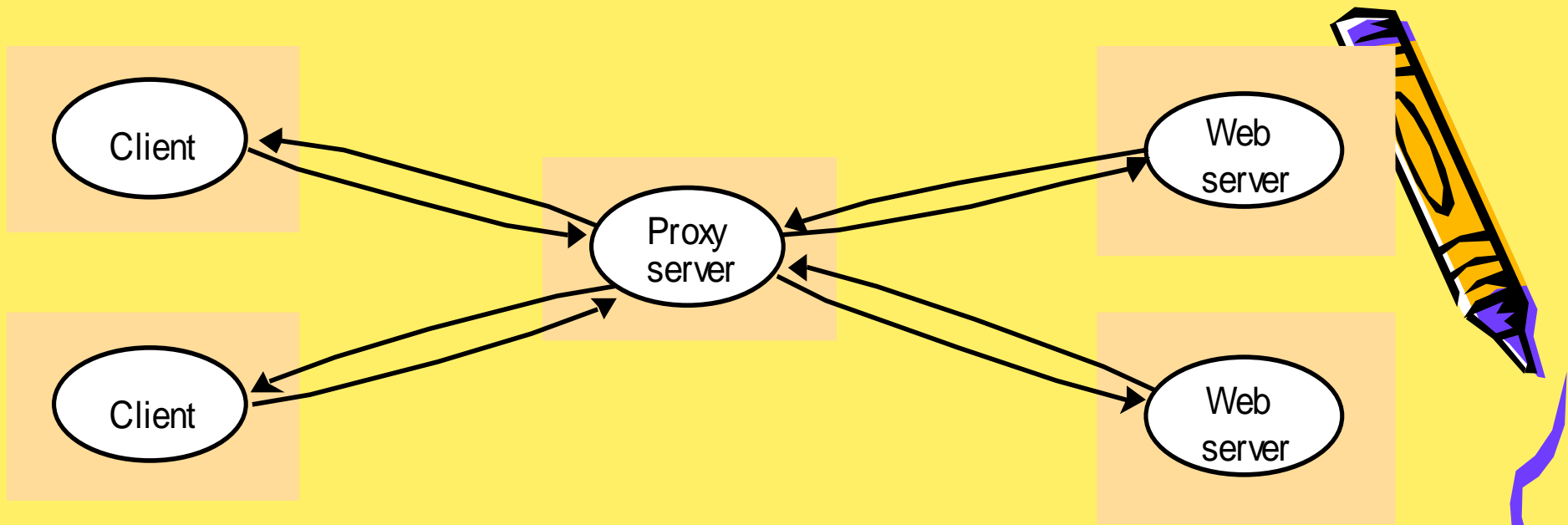
Multiple-Server



3-B Proxy Server

- Web proxy servers provide a shared cache of web resources for client machines at a site or across several sites.
- A cache is a store of recently used data objects (for example: recently visited web pages).
- When a new object is received at a computer it is added to cache store. (if it is full, the object will replace other objects).
- When an object is needed by a client process, the caching service first checks the cache and supplies it if it is available and up-to-date.
- If not, an up-to-date object is fetched.
- Caches is located at a proxy server that can be shared by several clients.



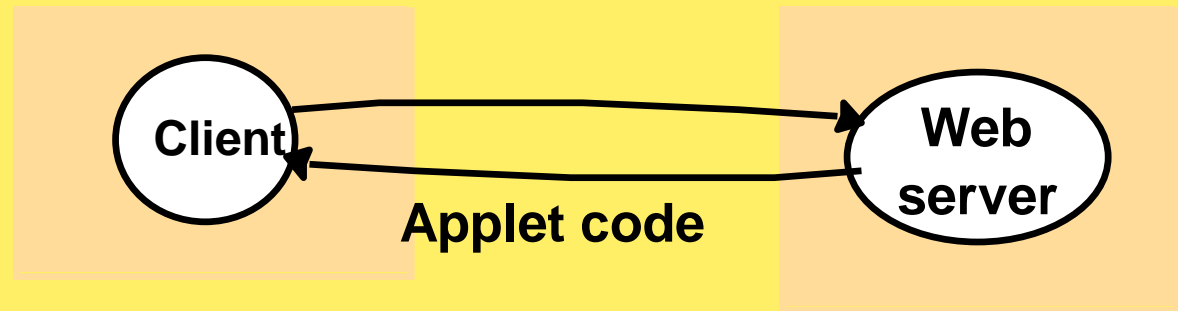


- The purpose of proxy servers is to increase availability and performance of services by reducing the load on WAN and web servers.
- Proxy servers can take on other roles, for example, they may be used to access remote web servers through firewall.

3-C Mobile Code



- An Example of mobile code is the applet (example Java)
 - The user running a browser selects a link to an applet whose code is stored on a web server.
 - The code is then downloaded to the user's browser and runs on the user's side.
 - Sometimes it is called code migration. Why?
 - Moving from heavily loaded to lightly loaded system (server and network).
- a) client request results in the downloading of applet code

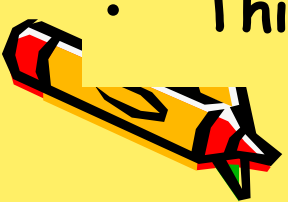




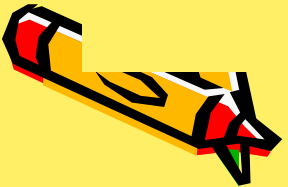
b) client interacts with the applet



Mobile Code Example: Applet

- Moving code to data, rather than data to code.
- Advantage of mobile code running locally on the user's side is that it gives good interactive response since it does not suffer from delays or variability of BW due to network variable loads.
- An example of an applet usage that we are familiar with is the up-to-date notifiers of new versions of services such as virus scan programs.
- The code in the applet automatically communicates the server and displays if there is new versions of the program.
- This is different than the normal way of interaction between the client/server as the client should have initiated the communication with the server each time.
- This type of interaction is called push model.



- 
- Another example, a stockbroker might provide his client with a service that automatically shows the current status of his shares.
 - Of course, the client has to download the special applet that receives the updates from the broker's server and display them for the client.
 - However, this rises the security problem !!!!!
 - As this applet is very dangerous to the local resources of the client's computer.
 - Therefore browsers give applets limited access to local resources such as not allowing the applet to access local files, printers or network sockets.
- 
- 

3-D Mobile agent

- A mobile agent is a running program (including code and data this time) that travels from one computer to another in a network carrying out a task on someone's behalf (client) and returning with the result to him.
- Examples of tasks are: collecting information or search.
- A mobile agent may use the local resources at each site visited (for example accessing a DB).
- If we compare this model with a static client that remotely invokes the resources that may include transferring data, then we will find a mobile agent converts the remote invocation into local invocation which means reduction in communication cost and time.

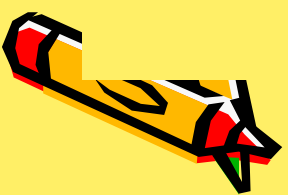




- Mobile agents can be used to:
 - - install and maintain SW on the computers within an organization.
 - - compare the prices of products from a number of vendors by visiting the site of each vendor and performing a series of DB operations.
- An old version of the mobile agent was presented at Xerox called worm program.
- It was designed to make use of the idle computers in order to carry out intensive computations.
- Just as the mobile code, the mobile agent rises the problem of security of the local resources on each visited computers.
- Thus, each of the visited computer that will receive the mobile agent should decide which local resource it should access according to the identity of the user initiating this agent.



- The user's identity must be included in a secure way with the code and data of the mobile agent.
- We have to note that a mobile agent may fail to complete its work if it is denied of access to resources it needs.
- For this reason, the applicability of mobile agents are limited.



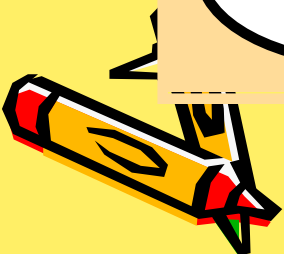
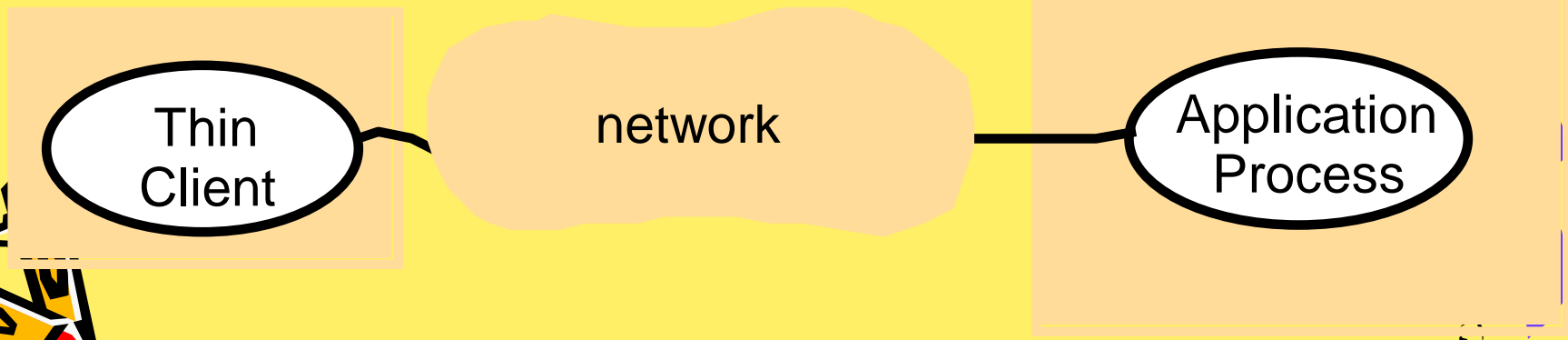
3-E Thin Client/Compute Server



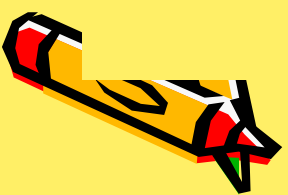
- Thin client = software layer that supports windows-based user interface on a computer who is local to the user while executing application programs on a remote computer.
- Instead of downloading the code of applications into the user's computer, it runs them on a compute server
- A Compute server is a powerful computer that has the capacity to run large numbers of applications simultaneously.
- It could be a multiprocessor or cluster computer.

Network computer or PC

Compute server



- The main disadvantages of this model is when it is used in highly interactive graphical activities (such as CAD and image processing) where the user suffers from delay due to the need to transfer image and vector information between the thin client and the application process at the server.

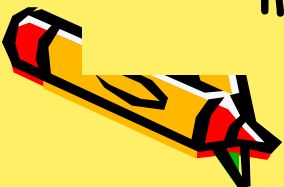


Types of Distributed Operating Systems



Network Operating Systems

- Such as Unix and windows. They have built-in network capabilities and so they can use it to access remote resources.
- Users are aware of multiplicity of machines.
- Unit of communication between the machines is the file
- Access to resources of various machines is done explicitly by:
 - Remote logging into the appropriate remote machine (telnet, ssh)
 - Remote Desktop (Microsoft Windows)
 - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism



Distributed Operating Systems

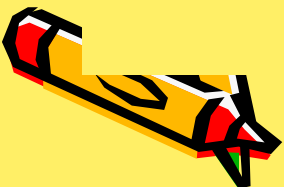
- Users are not aware of multiplicity of machines
 - Access to remote resources similar to access to local resources.

Different set of tasks provided by DOS:

- **Data Migration** – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task (i.e. Unit of communication between the processes is the file or part of file to the level of message)
- **Computation Migration** – transfer the computation, rather than the data, across the system
- **Process Migration** – execute an entire process, or parts of it, at different sites to achieve one of the following reasons:
 - Load balancing – distribute processes across network to even the workload
 - Computation speedup – subprocesses can run concurrently on different sites
 - Hardware preference – process execution may require specialized processor
 - Software preference – required software may be available at only a particular site
 - Data access – run process remotely, rather than transfer all data locally

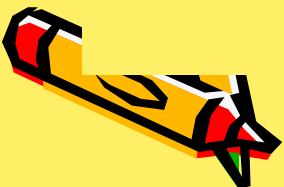
Communication Structure

- The design of a *communication* network must address four basic issues:
 1. Naming and name resolution - How do two processes locate each other to communicate?
 2. Routing strategies - How are messages sent through the network?
 3. Connection strategies - How do two processes send a sequence of messages?
 4. Contention - The network is a shared resource, so how do we resolve conflicting demands for its use?



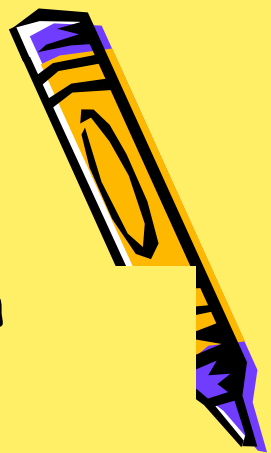
1 - Naming and Name Resolution in Network

- How two processes communicate in a network?
- We have many ways:
- Address messages with the process-id
- Identify processes on remote systems by
 <host-name, identifier> pair
- *Domain name service* (DNS) – specifies the naming structure of the hosts, as well as name to address resolution (Internet)



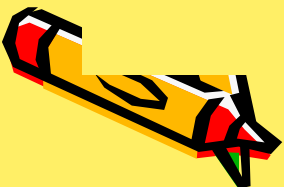
2- Routing Strategies

- Fixed routing - A path from *A* to *B* is specified in advance; path changes only if a hardware failure disables it
 - Since the shortest path is usually chosen, communication costs are minimized
 - Fixed routing cannot adapt to load changes
 - Ensures that messages will be delivered in the order in which they were sent
- Virtual circuit - A path from *A* to *B* is fixed for the duration of one session.
- Different sessions involving messages from *A* to *B* may have different paths
 - Partial solution to adapting to load changes
 - Ensures that messages will be delivered in the order in which they were sent





- Dynamic routing - The path used to send a message from site A to site B is chosen only when a message is sent
 - Usually a site sends a message to another site on the link least used at that particular time
 - Adapts to load changes by avoiding routing messages on heavily used path
 - Messages may arrive out of order
 - This problem can be solved by appending a sequence number to each message



3- Connection Strategies



- Circuit switching - A permanent physical link is established for the duration of the communication (i.e., telephone system)
- Message switching - A temporary link is established for the duration of one message transfer (i.e., post-office mailing system)
- Packet switching - Messages of variable length are divided into fixed-length packets which are sent to the destination
 - Each packet may take a different path through the network
 - The packets must be reassembled into messages as they arrive
- Circuit switching requires setup time, but incurs less overhead for shipping each message, and may waste network bandwidth
- Message and packet switching require less setup time, but incur more overhead per message



4- Contention

- Several sites may want to transmit information over a link simultaneously. Techniques to avoid repeated collisions include:
- CSMA/CD - Carrier sense with multiple access (CSMA); collision detection (CD)
 - A site determines whether another message is currently being transmitted over that link.
 - If two or more sites begin transmitting at exactly the same time, then they will register a CD and will stop transmitting
 - When the system is very busy, many collisions may occur, and thus performance may be degraded
- CSMA/CD is used successfully in the Ethernet system, the most common network system





- Token passing - A unique message type, known as a token, continuously circulates in the system (usually a ring structure)
 - A site that wants to transmit information must wait until the token arrives
 - When the site completes its round of message passing, it retransmits the token
 - A token-passing scheme is used by some IBM and HP/Apollo systems
- Message slots - A number of fixed-length message slots continuously circulate in the system (usually a ring structure)
 - Since a slot can contain only fixed-sized messages, a single logical message may have to be broken down into a number of smaller packets, each of which is sent in a separate slot
 - This scheme has been adopted in the experimental Cambridge Digital Communication Ring

